УДК 681.3

СЕРГІЄНКО А.М.

СИНТЕЗ СТРУКТУР ДЛЯ ВИКОНАННЯ ПЕРІОДИЧНИХ АЛГОРИТМІВ З ОПЕРАТОРАМИ КЕРУВАННЯ

Запропоновано метод відображення алгоритму, заданого умовним графом синхронних потоків даних, в конвейерний обчислювальний пристрій, що описаний на мові VHDL. Приведено приклад синтезу пристрою для обчислення модуля комплексних чисел, який реалізовано в програмованій логічній інтегральній схемі з високою пропускною спроможністю і малими апаратними витратами.

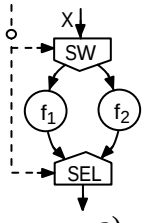
A method of mapping of Boolean SDF graph into pipelined datapath which is described by VHDL program. A synthesis example of the complex number magnitude calculator illustrates the method. This calculator is configured in FPGA and provides both high throughput and small hardware volume.

До періодичних алгоритмів належать алгоритми, що виконують один і той самий алгоритм з повторенням для різних груп вхідних даних. До них належать алгоритми цифрової обробки сигналів, керування, вирішення задач лінійної алгебри, тощо. Такі алгоритми прийнято задавати у вигляді графів потоків даних. Використання графа синхронних потоків даних (ГСПД) дає змогу формально будувати на його основі оптимальну обчислювальну схему зі статичним розкладом виконання алгоритму [1]. В роботах [2,3] запропоновано метод відображення ГСПД, представленого в просторі ресурси – час у вигляді конфігурації алгоритму (КА), в структуру обчислювального пристрою на рівні регістрових передач. Суть метода полягає в тому, що актор графа – оператор алгоритму, якому відповідає вектор $\boldsymbol{K}_{p,q}$ в КА – відображається в p-у вершину процесорного елемента (ПЕ) і в деякий такт свого виконання. В даній статті цей метод вдосконалений, щоб відображати алгоритми з операторами керування.

Для представлення і відображення алгоритмів з умовними розгалуженнями в роботі [4] були запроваджені умовні ГСПД (Boolean, conditional SDF graphs), які відрізняються від звичайних ГСПД тим, що вони мають декілька паралельних (альтернативних) гілок (див. рис. 1,а). Ці гілки починаються в вершині перемикача SW і сходяться в вершині селектора SEL, причому одна з них активізується тільки тоді, коли актор умови T є істинним а інша – коли хибним.

При інтерпретації ГСПД, тобто при моделюванні алгоритму, кожна вершина активізується, якщо на її входах є певна кількість токенів, що асоціюються з даними, після чого вона видає на свої виходи токени – результати. Статичний розклад для ГСПД можна скласти, якщо граф є узгодженим, тобто якщо при його інтерпретації кількість токенів на дугах залишається постійною і не виникає блокування. В роботі [4] доведено,

що в загальному випадку за жений час перевірити те, що умовний ГСПД є узгодженим, неможливо.



а)

```
if T=true then
    Y= f1(X);
else
    Y= f2(X);
end if;
```

б)

Рис.1. Фрагмент умовного ГСПД (а) і відповідний йому умовний оператор (б)

Квазістатичним називається такий розклад ГСПД, коли момент запуску деякого актора знаходиться динамічно, тобто під час інтерпретації ГСПД, але запуски решти акторів можна знайти за статичним розкладом. В роботі [4] показано, що для умовного ГСПД, у якого альтернативні гілки відповідають оператору if-then-else (див. рис.1,б), або оператору case з усіма можливими комбінаціями умови селектора, можна скласти квазістатичний розклад. При цьому, наприклад, моменти запусків акторів f_1, f_2 (рис.1,а) знаходяться динамічно в залежності від токена T, а запуск актора SEL знаходиться з урахуванням максимальної з затримок виконання акторів f_1, f_2. Таким чином, враховуючи вищеназвані особливості умовного ГСПД, можна будувати КА з умовними операторами, що буде коректно відображатись в структуру обчислювального пристрою (ОП) і його розклад роботи.

Умовний ГСПД з квазістатичним розкладом взаємно однозначно відповідає деякій VHDL–програмі, в якій та чи інша гілка алгоритму вибирається за оператором if-then-else або case. При цьому можна наполягти, щоб логічна умова цих операторів не була константою, або виразом, що спрощується до константи, а в гілках операторів були розглянуті всі без винятку логічні альтернативи. При таких умовах одержимо узгоджений умовний ГСПД, як і в [4]. При відображенні програми на VHDL в умовний ГСПД слід дотримуватись таких самих правил відображення, що представлені в [3]. Доповненням є те, що вершиною перемикача є вершина, яка генерує відповідний операнд, альтернативам в операторах if-then-else, case ставляться у відповідність вершини операцій, що виконуються при визначених умовах, а фразі end if або end case відповідає вершина селектора, в яку зливаються альтернативні гілки.

При синтезі ОП з опису на мові VHDL оператори if-then-else або case можуть відображатися з розділенням ресурсів [3]. Якщо при відображенні умовного ГСПД оператори присвоювання, що стоять в альтернативах логічних операторів будуть розділяти апаратні ресурси, то по-перше, можна одержати статичний розклад, по-друге, апаратні витрати в результуючому ОП будуть мінімізованими. Для цього доведемо наступне твердження.

Твердження. Якщо в умовному ГСПД підграфи умовних розгалужень відповідають операторам if-then-else або case, в гілках яких розглянуті всі без винятку логічні альтернативи, які відображаються в єдиний апаратний ресурс, то для такого ГСПД можна скласти статичний розклад.

Доведення. Виходячи зі згаданого вище, даний ГСПД має квазістатичний розклад. Якщо крім цього можна визначити моменти запуску акторів для кожної з альтернативних гілок до інтерпретації ГСПД, то тоді розклад буде статичним. Дійсно, за умовою, коли всі без винятку актори в альтернативних гілках відображаються в єдиний апаратний ресурс, всім цим акторам можна призначити один і той самий момент запуску. При цьому вершина селектора буде вибирати результат того актора, для якого токен-селектор буде істинним. Через те, що в альтернативних гілках розглянуті всі без винятку логічні альтернативи, при будь-якому значенні токена-селектора на відповідному вході вершини селектора завжди буде з"являтись рівно один токен результату. Це є умовою того, що в ГСПД не виникає ні блокування, ні накопичення токенів у вхідних дугах вершини селектора, тобто ГСПД є узгодженим. Так як актори в альтернативних гілках відображаються в єдиний апаратний ресурс, тобто ПЕ, а не в два або більше ПЕ, то це є умовою мінімізації апаратних витрат.

Розглянемо метод синтезу структур ОП для виконання періодичних алгоритмів з операторами керування, що оснований на методі відображення КА в структуру конвейєрного ОП. Начальними даними для синтезу структури ОП приймаються умовний ГСПД алгоритму та період його виконання L. Процес одержання ефективної конфігурації алгоритму має три етапи.

На першому етапі вершини-оператори ГСПД разом з дугами розташовуються в трьохвимірному просторі як множини векторів $\boldsymbol{K}_i$ та $\boldsymbol{D}_j$ з урахуванням умов, приведених в [2,3]. При цьому виконується мінімізація числа ПЕ шляхом виконання вимог $|\boldsymbol{K}_{p,q}| \to L$, тобто число вершин, що відображаються в один ПЕ, прямує до L. Крім того, при формуванні КА бажано будувати досконалий кістяк ГСПД, як це пропонується в [5].

При розташуванні логічних операцій, вершині селектора ставиться у відповідність вершина-оператор мультиплексора, до входів якого під'єднуються кінці альтернативних гілок. Цей мультиплексор керується сигналом, що приходить по керуючій дузі з вершини, що генерує логічний сигнал. Початки альтернативних гілок під'єднуються до відповідних вершин – джерел даних для цих гілок.

На другому етапі мінімізуються оператори в альтернативних гілках. Так як згідно з вищеприведеним твердженням оператори з альтернативних гілок відображаються в один і той самий ПЕ, а також виконуються в одному такті, то вони повинні мати однакові координати, тобто їх можна зклеїти в одну вершину. При цьому вершина оператора після зклеювання повинна мати входи як для операндів, так і для керуючого сигнала, що перемикає функції оператора у відповідності з виконанням альтернативних гілок, а вершина мультиплексора альтернатив зникає.

Слід відмітити, що розділяти один і той самий ресурс можуть не тільки вершини-оператори, що стоять в одному ярусі альтернативних

гілок (з однаковими часовими координатами), а і взяті з різних ярусів. Більш того, це можуть бути вершини-оператори, що належать альтернативним гілкам з різних підконфігурацій КА. Але умови виконання цих гілок не повинні перекриватись, щоб не було випадків, коли вершини, що зливаються, можуть виконуватись одночасно. Треба додати, що вказані випадки злиття вершин є можливими, але можуть бути недоцільними через те, що для перетвореної КА не можна буде скласти ефективний розклад. Також злиття вершин може не привести до зменшення апаратних витрат, якщо ці вершини відповідають різнорідним операціям.

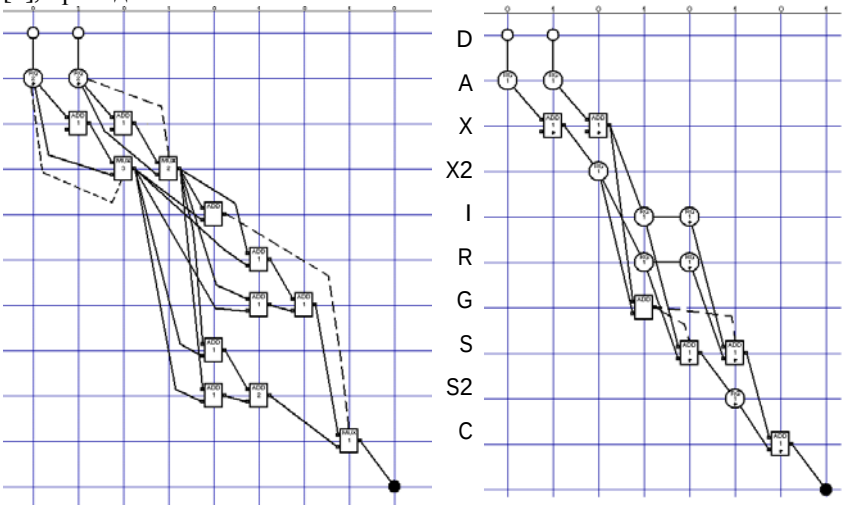
На третьому етапі виконується врівноважування КА. Після цього виконується оптимізація КА шляхом взаємних перестановок векторів-вершин з одного ярусу з метою мінімізації числа регістрів та числа входів мультиплексорів в результуючій структурі і/або з застосуванням інших стратегій, наприклад, ресинхронізації [3,6]. Крім того, положення вершин-мультиплексорів може бути оптимізоване таким самим чином, як і положення інших вершин, вихідні дуги з них можуть мати довільну часову складову, в тому числі і нульову. В останньому випадку вони не відображатимуться в мультиплексор з регістром і їх можна буде мінімізувати в структурі ОП шляхом злиття з вхідним мультиплексором ПЕ, що стоїть після нього. Одержана оптимізована КА відображається в граф структури ОП шляхом склеювання векторів-вершин з одинаковими координатами *k, l*. КА перетворюється в розклад виконання операторів, використовуючи ту властивість, що момент виконання оператору, що відповідає вектору $\boldsymbol{K}_i = \langle k_i, l_i, t_i \rangle$, дорівнює t_i. При цьому можна не будувати структуру і розклад, якщо зразу описати ОП на мові VHDL [3].

Розглянемо застосування методу на прикладі синтезу ОП для обчислення модуля комплексного вектора $c = |a+jb|$ за алгоритмом Понселе [7]. Цей алгоритм полягає в кусково-лінійній апроксимації функції двох змінних в залежності від відношення амплітуд значень *a,b* і може бути представлений наступною VHDL-програмою:

```
    entity MAGNITUDE is
        generic(n: integer:=16);        – розрядність даних
        port (    A: in INTEGER range -2**n to 2**n-1;
                  B: in INTEGER range  -2**n to 2**n-1;
                  C: out INTEGER range  0 to -1+2**n+ 2**n);
end MAGNITUDE;
architecture MAGN_BEH of MAGNITUDE is
        signal Z,R,I : integer          range -2**n to 2**n-1:=0;
    begin
        process(A,B) begin
                  if A<0 then  – обчислення абсолютної величини операндів
                            R<=-A; – в залежності від їхніх знакових розрядів
                  else
                            R<=A;
                  end if;
```

```
            if B<0 then
                    I<=-B;
            else
                    I<=B;
            end if;
            if R<I then              – визначення більшого з операндів
              C<=(I + R/4) -(I/32-R/8);    – апроксимація по Понселе
            else  – ділення на 4,8,32 виконується шляхом зсуву даного
              C<=(R + I/4) -(R/32-I/8);
        end if;
    end process;
end MAGN_BEH;
```

Нехай вхідні дані *a* і *b* приходять по одній вхідній шині послідовно і алгоритм виконується з періодом $L = 2$ такти синхросерії CLK в конвейєрному режимі. На першому етапі будується КА, що відповідає цьому алгоритму, яка показана на рис. 2,а. Слід відмітити, що на рисунку дуги, що відповідають керуванню мультиплексорами, зображені пунктиром, а зсув операндів, тобто ділення на 4,8,32 – не відображений. На другому етапі виконується розділення ресурсів вершин-суматорів. І нарешті, після третього етапу будується урівноважена оптимізована КА, що показана на рис. 2,б. Опис результуючого ОП, одержаний з КА за методикою [3], приведено нижче.



а) б)

Рис.2. Начальна (а) і кінцева (б) конфігурація алгоритму Понселе

```
entity MAGNITUDE2 is generic(n: integer:=16); – розрядність даних
    port (CLK: in BIT;
          RST:in BIT;                          – начальне встановлення
          D: in INTEGER range -2**(n-1) to 2**(n-1)-1;
```

```
                C: out INTEGER range  0 to -1+2**n);
end MAGNITUDE2;
architecture MAGN_SYNT of MAGNITUDE2 is
        signal X,X2,R,I,A,S,S2 : integer range -2**n to 2**n-1:=0;
        signal ct: bit:='0';
        signal G:boolean:=false;
   begin
  CT2:process(CLK,RST) begin  – лічильник тактів
            if CLK='1' and CLK'event then
                    if RST='1' then              ct<='0';
                    else  ct<=not ct ;
                    end if;
                end if;
            end process;
MAIN:process(CLK) begin
            if CLK='1' and CLK'event then
                    A<=D; – вхідний регістр
                    if A<0 then
                       X<=0-A;    – абсолютна величина
                    else
                       X<=A;
                    end if;
                    case ct is  – те, що виконується в парних і непарних тактах
                    when '0' =>X2<=X;     –  парний такт
                            if G = true then
                                S<= R + I/4;
                            else
                                S<= R/4 +I;
                            end if;
                            C<=S2 – S;          – регістр результата
                    when others=>  I<=X;   – непарний такт
                            if (X2 >X) then
                                G<= true;
                            else
                                G<=false;
                            end if;
                            R<=X2;
                            S2<=S;
                            if G = true then
                                S<= R/32-I/8;
                            else
                                S<= I/32-R/8;
                            end if;
                        end case;
                end if;
            end process;
    end MAGN_SYNT;
```

Результуючий ОП MAGNITUDE2 для 16-розрядних даних компілятором-синтезатором Synplify перетворюється в схему на рівні вентилів для

ПЛІС Virtex-4 фірми Xilinx з апаратними витратами 122 LUT, яка може виконувати розрахунки в конвейерному режимі з тактовою частотою до 200 МГц. При включенні режиму ресинхронізації компілятор Synplify дає схему, що працює з частотою до 235 МГц. Для порівняння, ОП MAGNITUDE, що виконує начальний алгоритм, після конфігурації в ПЛІС має апаратні витрати 219 LUT і тактову частоту лише 100 МГц. В порівнянні з аналогічним ОП, що також виконує алгоритм Понселе [7], цей ОП має більшу точність (менше 4%) і побудований за формальним методом.

В даній статті запропоновано метод синтезу структур для виконання періодичних алгоритмів з операторами керування, що є удосконаленням методу відображення графу синхронних потоків даних, представленого у вигляді конфігурації алгоритму. Метод дає змогу формально відображати алгоритми з операторами керування в структуру конвейерних обчислювальних пристроїв з заданим періодом обчислень, що мають мінімізовані апаратні витрати і високу тактову частоту. При цьому через те, що результатом є опис ОП на VHDL, метод дає змогу не будувати власне структуру ОП і розклад виконання алгоритму, а перекласти це завдання на компілятор-синтезатор. Наведений приклад синтезу ОП для обчислення модуля комплексного числа доводить високу ефективність метода.

Список посилань

1. Bhattacharyya S.S., Leupers R., Marwedel P. Software Synthesis and Code Generation for Signal Processing Systems // IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing, – 2000. –V47. –№9. –p.849-875.
2. Каневский Ю.С., Овраменко С.Г., Сергиенко А.М. Отображение регулярных алгоритмов в структуры специализированных процессоров // Электрон. Моделирование.–2002.–Т.24.–№2.–С. 46-59.
3. Сергиенко А.М. VHDL для проектирования вычислительных устройств. – Киев: Диасофт. –2002. – 210 с.
4. 4.J. T. Buck. Scheduling Dynamic Dataflow Graphs with Bounded Memory using the Token Flow Model. PhD thesis, Dept. El. Engineering and Computer Sci., Berkeley Univ. -1993.-173p.
5. Сергієнко А.М. Досконалий кістяк графа алгоритма// Вістник НТУУ"КПІ". Інформатика, управління та обчислювальна техніка. -2007. -№46. –с.62-67.
6. Сергиєнко А.М., Симоненко В.П. Отображение периодических алгоритмов в программируемые логические интегральные схемы //Электронное моделирование. –Т.29. -2007. -№2. –с.49-62.
7. Каневский Ю.С., Клименко М.К., Сергиенко А.М. Устройство для вычисления функции $\sqrt{Y^2 + X^2}$. А.С. СССР N 1541601, Б.И. №5, 1990.